NASA Technical Memorandum 100573

THE ENVIRONMENT FOR APPLICATION SOFTWARE INTEGRATION AND EXECUTION (EASIE) VERSION 1.0

VOLUME I

EXECUTIVE OVERVIEW

LAWRENCE F. ROWELL
JOHN S. DAVIS

MARCH 1989

(NASA-TM-100573) THE ENVIRONMENT FOR APPLICATION SOFTWARE INTEGRATION AND EXECUTION (EASIE) VERSION 1.0. VOLUME 1: EXECUTIVE OVERVIEW (NASA) 41 p CSCL 09B

N89-21538

Unclas
G3/61 0198843

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

PREFACE

The Environment for Application Software Integration and Execution (EASIE) provides both a methodology and a set of software utility programs to ease the task of coordinating engineering design and analysis codes. The need for such techniques and tools has stemmed from the computer-aided design and engineering activities within Langley Research Center's Space Systems Division (SSD). In SSD, the Vehicle Analysis Branch (VAB), with emphasis on advanced transportation systems, and the Spacecraft Analysis Branch (SAB), with emphasis on advanced spacecraft, share a common need to integrate many stand-alone engineering analysis programs into coordinated, quick-turnaround, user-friendly design systems. In particular, the most needed capabilities include easy selection of application programs, quick review and modification of program input/output data, and ability to log the actual steps that were executed during a study. Although the application programs used by VAB and SAB differ, the design methods used by their engineers are quite similar, and great efficiency can be gained by providing a computer environment with the capabilities mentioned above.

EASIE is a user interface and set of utility programs which supports rapid integration and execution of programs about a central relational database. In general, the EASIE system addresses the needs of four different classes of people who will be involved in the development of an engineering design system.

Certain individuals may serve in more than one of these roles, but the following terms will help to clarify several distinct activities associated with the EASIE system.

The first classification represents the engineer/designer/analyst. This group conducts the design study by executing modeling and analysis programs and generating data required to evaluate the design against its objectives. EASIE documentation will refer to this group as EASIE system users or, more often, as users. In general, these users are only interested in executing programs already installed into an EASIE design system.

A second group aided by EASIE is identified as application programmers. These programmers are responsible for the development and improvement of modeling and analysis programs used in the engineering design process. They are the experts with respect to particular application programs who can define the input and output variables. This must be done before inclusion of that program with others into the integrated system.

The third group is identified as program implementers, since their function is to provide an environment where all the software tools work together with a minimum of effort. These people will use information provided by the application programmers and will install or modify the programs in an EASIE system by creating appropriate data constructs in the database and locating files where needed by the EASIE executive.

The fourth classification is design team leader or design manager. This is the individual or group responsible for

identifying parameters important to the design study and for configuration management of the data as it is produced by the design team. This design manager must have an overview of the total data requirements for the analysis process and the foremost concern for the integrity of the data.

With these terms defined, the four volumes of EASIE documentation can be associated with the groups most likely to use them. Each of the volumes addresses different aspects of the support tools, and each is intended to be independent of the others.

Volume I, EXECUTIVE OVERVIEW, provides information about the functions, concepts, and historical development of EASIE and should be read by anyone trying to determine if EASIE would be beneficial to his or her work.

Volume II, PROGRAM INTEGRATION GUIDE, describes the portion of the EASIE tools supporting both the integration of application programs into a central database and the definition of the data dictionary used during data review and modification. This volume will be used primarily by the program implementer and the design manager in their responsibilities for the actual installation of appropriate programs into a fully integrated design system. However, the application programmer may also use tools described in this volume to assist in the documentation of input/output variables for the application program.

Volume III, PROGRAM EXECUTION GUIDE, describes the portion of the EASIE tools supporting the selection and execution of application programs, building of menus, and editing of program

data. This volume will be of foremost importance to the engineer/designer/analyst who will perform design studies. In addition, the program implementers will find the sections concerning the construction of application-dependent procedures helpful. Finally, this document will also be used by design managers for reviewing data and design activities.

Volume IV, SYSTEM INSTALLATION AND MAINTENANCE GUIDE, describes the procedure of loading the EASIE system onto a computer. It also gives some insight into the hardware and software dependencies of the EASIE code. This, most likely, will be needed by the program implementer for familiarization with the directory structure and location of the various EASIE components. Although the design of EASIE is intended to reduce the system dependencies, this version nevertheless reflects in several ways the current implementation using the Relational Information Management (RIM*) database management system and the VAX/VMS+ operating system.

* Trademark of Boeing Computer Services

\+ Trademark of the Digital Equipment Corporation

ORIGINAL PAGE IS
OF POOR QUALITY

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

The increasing complexity of today's aerospace vehicles requires ever more sophisticated design approaches and analysis techniques which can only be realized by computer-aided engineering systems. In order to accomplish the many iterations required of the designer, these complex analysis codes must be coordinated to provide user-friendly, quick-turnaround, computer-based design systems. The Environment for Application Software Integration and Execution (EASIE) provides a methodology and set of utility routines for a design team to build, maintain, and apply computer-aided design systems consisting of large numbers of diverse, stand-alone analysis codes. The EASIE approach uses a central database containing all the variables (the data dictionary) needed as inputs to the analysis programs that will make up the integrated system. Utilities exist for constructing the data dictionary and database schema, generating the subroutines to read from or write to the database, incorporating the analysis programs into the database, interactively reviewing and modifying values in the database, incorporating the analysis programs into an interactive executive for selection and execution, building menus and procedures to assist the user, logging activities of the analyst, and managing the data configurations that evolve during the design process. This document gives an overview of the functions provided by EASIE and describes their use. Three operational design systems based upon the EASIE software are briefly described.

2.0 BACKGROUND

Over the past 25 years, the application of computers in engineering has increased exponentially. Today, many steps in engineering design and analysis can now be completed in hours that ten years ago would have taken weeks to accomplish. As the computer environment (processors, support software, workstations, communications) has increased in capability, so have the analysis techniques and programs for the engineering disciplines. However, this growth has not been without a price. While information may be obtained at a faster rate and in greater abundance, the absence of standardization and lack of coordination among software developers have resulted in the proliferation of computer programs that are unable to communicate data easily to other programs. The result is often the manual transfer of data from one program to another with the associated manpower loss, time delay, and potential error introduction. Therefore, the efficiency with which design studies can be accomplished is often more dependent on the coordination of the analysis programs and their data interchange requirements than on the processing speed of the computer system.

In typical design and engineering studies, use will be made of both commercially and in-house developed programs, so that the purchase of integrated analysis tools still leaves at issue the coordinaton of the commercial database with the local programs. This problem is particularly acute for conceptual design studies which are often characterized by advanced technologies requiring new analysis codes and by numerous competing configurations

requiring many analysis iterations and refinements. The constant change in design methods does not allow the system tools to reach the maturity that is possible in production situations.

The Space Systems Division (SSD) at NASA's Langley Research Center (LaRC) has been addressing this problem area since the early 1970's. In SSD, the Vehicle Analysis Branch (VAB), with emphasis on advanced transportation systems, and the Spacecraft Analysis Branch (SAB), with emphasis on advanced spacecraft, share a common need to integrate many stand-alone engineering analysis programs into coordinated, quick-turnaround, user-friendly design systems. In particular, the most needed capabilities include easy selection of application programs, quick review and modification of program input/output data, and ability to log the actual steps executed during a study. Although the application programs used by VAB and SAB differ, the design methods used by their engineers are quite similar, and great efficiencies can be gained by providing a computer environment that yields the capabilities mentioned above.

Wilhite [1] described the development history of several integrated design systems used for launch vehicle design and gave an overview of the capabilities needed in future enhancements. The Environment for Application Software Integration and Execution, EASIE, implements these needed capabilities and provides both a methodology and a set of software utility programs to ease the task of coordinating engineering design and analysis codes. EASIE consists of a user interface and a set of utility programs which support rapid integration and execution of programs about a central relational database.

3.0 THE EASIE PROGRAM APPROACH

The purpose of the EASIE is to aid in the complex task of integrating a collection of application programs into a single system. The tools offer a program integration approach critical in defining and forming the data paths for inter-program communication. Although the acronym EASIE implies simplicity, the problems encountered when integrating two or more programs are often formidable. Consequently, the potential EASIE user should anticipate a relatively high learning curve upon initial exposure to the database management system (DBMS) terminology and software tools. However, one of the principal advantages of using the EASIE tools is that once the initial learning curve is climbed, the learned program integration techniques are applicable to other programs or sets of programs.

In addition to understanding program integration problems (addressed below), some experience with DBMS terminology and techniques is necessary. In particular, the EASIE tools rely heavily on the techniques of the relational approach to DBMS such as those described by C. J. Date [2]. More specifically, the current version of the EASIE tools is built in FORTRAN 77 based upon the Relational Information Management [3] (RIM)* DBMS application program interface and executes on Digital Equipment Corporation VAX+ computers.

* A trademark of Boeing Computer Services

\+ A trademark of the Digital Equipment Corporation

ORIGINAL PAGE IS
OF POOR QUALITY

A generalization of current program coupling techniques is given to show the basis for the EASIE program approach. There are a number of techniques currently being used to couple independent analysis programs into design systems, and these are generalized here in four categories: close-coupled integration, loose-coupled integration, close-coupled interfacing, and loose-coupled interfacing. The integration technique uses direct communication between a program and a central database or another program. Interfacing means indirect data communication among programs that operate in the normal input-output file mode using intermediary programs to properly format the data. Close-coupling means that the path through the analysis techniques or programs is fixed. With loose-coupling, the programs can be individually executed, or execution paths can be programmed.

Close-coupled integration (Figure 1a) is analogous to one large program where the analysis is performed by modules (subroutines, segments, or overlays) and data transfer is accomplished by global common and data files. The advantage of close-coupled integration is the speed of execution because data communication is direct. The disadvantages of close-coupled systems are, first, the difficulties of integrating all the separate analysis codes into one computer program and, second, of adapting to unique configurations requiring new analysis techniques because these techniques are deeply rooted into the program and are not easily changed. Loose-coupled integration (Figure 1b) is the technique employed by most business-oriented systems today. A central database management system is used for

data communication to all the separate programs. The main advantage of loose-coupled integration is that the programs (analysis techniques) can be developed independently and can later be coupled together to form a complete design system. The disadvantage of the loose-coupled integration is that these programs must be written to communicate with the database either initially when the program is developed or later after completion of the program. For a program that has been developed independently of any database considerations and has a large data input with many analysis options, it is an awesome task to integrate the database communication code into the program unless some support tools and automated procedures are developed to simplify the task.

Close-coupled interfacing (Figure 2a) is one of the first techniques used for coupling independent analysis programs. The interfacing starts with the coupling of one program to another by having the first program write an input file for the next program that is identical to the input file the program used before coupling. This process is repeated from program to program and eventually creates a network of closely coupled analysis programs that comprise a design system. As the network of programs grows, it becomes more and more difficult to couple new programs into the system. A new program may require input from several programs that are not directly linked. A separate program, called an intermediary program, must be written to read the output of several programs and create an input file for this new analysis program. Because of this problem of decentralized

information and because the path of the analysis is predefined by the program network, the current approach is to centralize the data. Loose-coupled interfacing (Figure 2b) uses a central database as a repository of data, but a pre-processor program must be written to retrieve, transform, and format the data into an input file that the program expected before coupling. After program execution, analysis results are written to the database by adding database code to the analysis program or by creating a post-processor to read output files and place appropriate data in the database. The advantage of interfacing with a pre-processor program is that it requires no knowledge of the internal coding of the analysis program and can be used when program source code is not available as with many commercial programs. Only the program input requirements are needed which are usually well documented. The disadvantages of interfacing are that a pre-processor program must be written for each analysis program and that there is computer overhead for writing an input file and reading that file by the analysis program.

Loose-coupled integration has been selected as the approach of choice, but it should be noted that the EASIE tools will equally support loose-coupled integration and loose-coupled interfacing requirements. The advantages of loose-coupling are: (1) the ease of incorporating new programs by using a central database for communication; (2) analysis programs can be executed as needed to aid the design process, or several can be executed for multidiscipline analyses; and (3) depth of analysis can be changed as the configuration matures. The system

architecture represented by Figure 1(b) also allows the development of a single, standard data editor/reviewer when appropriate constructs are created. Once programs are satisfactorily integrated around a common database, a flexible method is needed for interactively accessing the chosen program, gaining help information on demand, reviewing and modifying the input data, executing the program, and examining the results. These activities can obviously be performed within the computer operating system by using its native command language and editors, but such an approach does not offer any guidance to the designer, nor does it provide any specific framework for creating or tracking design procedures, i.e., sequences of program executions intended to optimize or iteratively arrive at a design goal. Again, the EASIE tools provide a powerful and flexible executive which presents the user a viewpoint within the integrated design system where either menus or a simple command language can perform activities typical in design studies. Section 4 describes the functions provided by the EASIE tools to enable construction of loose-coupled integrated systems of programs, a common editor, and a powerful executive.

ORIGINAL PAGE IS
OF POOR QUALITY

4.0. THE EASIE FUNCTIONS

This section briefly describes the capabilities that must be provided by support software if it is to fill the need for efficient database definition, program integration, and program execution. The following discussion of the capabilities will assume that a central database will be used to store information for the design activity and to communicate data among programs. Although computer data files are often used instead of database management systems (DBMS) to interface programs, these do not provide the versatility of the DBMS approach utilized by EASIE.

In order to describe how EASIE tools have been implemented to meet the needs examined in section 3, some concepts must be described. As each term or concept is discussed, its role in the EASIE system will be explained.

4.1 Dictionary - Data Definition

The EASIE database serves not only as a repository for the value of a variable, but it also contains essential descriptive information about the variable in the form of a data dictionary. Specifically, the data dictionary contains the variable name, its engineering units, a display format, a textual definition, and certain language-specific characteristics needed to support data input and output for any particular language (FORTRAN, Pascal, etc.). This information, placed in the database during its construction, can be called on whenever a variable is being reviewed for modification. In addition, information contained in the data dictionary is critical when

deciding upon the similarity of variables required by two different programs. This data dictionary structure is one of the key features of EASIE and is integral in providing many functions described later.

The information required for the data dictionary usually can only be provided by the program experts, or application programmers who are intimately familiar with the codes being integrated or interfaced. Although there are no shortcuts to this step in the integration process, aids have been devised to guide the application programmers in developing this information. An EASIE coding form, Figure 3, provides a simple and orderly vehicle for the collection and recording of the information on each variable that is needed to build the data dictionary. Also, an interactive dictionary-building processor, which prompts for the variable information, may be used by the program implementer during the actual creation of the database architecture or schema.

4.2 Schema - Data Structure

Minimally, for each program that is integrated into the database, all program input variables and any program output variables to be shared with other programs must be described in the data dictionary. It is acceptable, but unnecessary, to describe unshared output variables in the data dictionary for future uses. The program implementer, using information provided by the applications programmers, is responsible for logically grouping the identified data dictionary variables into relations

or tables which collectively are called the schema. The schema is simply the architecture of the total database. The relation is a DBMS term referring to a construct in the database. No further discussion of relations or schema need be given here, but more information can be obtained from the RIM documentation [3] or EASIE Volume II [4]. The importance of the program implementer in identifying and systematically grouping variables into relations for the data dictionary cannot be overemphasized, since this greatly influences the ease of maintenance of the database. Therefore, the program implementer should have a thorough knowledge of the program to be integrated and the other programs already integrated into the system. No software tools can substitute for this knowledge, but an interactive menu-driven processor, called MAKDICT, is the dictionary builder provided to aid the program implementer in the actual construction of the data dictionary. The facilities of MAKDICT are detailed in reference [4], and Table I provides a summary of its functions. The MAKDICT functions supporting the dictionary-building activity are: build relations (BR), add variables to a relation (AR), build a database schema dump (BS), list relations (LR), and print a relation (PR). Also, MAKDICT functions are used to connect each program to the database by generating appropriate database read and write routines for the specific variables required by each program and to provide utilities for data review and modification. The method of identifying which variables in the database are needed by each program leads to the concept of a data template.

4.3 Template - Data Selection

Once the data dictionary has been constructed for all variables of interest, a means must be provided to specify which variables in the database are to be associated with any particular program. Obviously, variables may be inputs and/or outputs for more than one program. The relationships between programs and their input and output variables are described by a template. A program input template is simply a list of the input variables (actually, both the variable name and the relation in which it is stored) that are required for a particular program. Likewise, a program output template is a list of the outputs (variable names and relations) which are to be inserted into the database by a particular program. These templates, once built, are associated by name with the particular application program to which they relate. This concept of program input and output templates enables the development of a single generic interactive editor which can display the input or output variables for any selected program simply by requesting the appropriate template. The full set of data dictionary information is presented by this editor, known as the REVIEWER and described in section 4.4. In addition, these templates enable the automatic generation of FORTRAN subroutine source code by the FORMATTER processor. The generated code is placed in each program to retrieve data from or store data into the database during program execution. The MAKDICT functions supporting the creation and printing of the templates and the generation of the FORMATTER subroutines are: build a template (BT), list a template (LT), print a template

ORIGINAL PAGE IS
OF POOR QUALITY

(PT), and build FORMATTER subroutines (BF). There are times when selected sets of variables in the database will be of interest, and they are unrelated to any particular program. Templates can be built that group any set of variables together to define any view of the database of interest for observation or transfer.

4.4 REVIEWER - Data Manipulation

To this point, means have been described for constructing a database structure, the associated data dictionary, and templates. Obviously, the engineer preparing to execute an analysis program is primarily interested in reviewing and editing the set of data inputs for his program and reviewing the program output results of his study. For this purpose, a data REVIEWER has been provided. The REVIEWER program is an interactive editor that presents a standard display, regardless of the application (analysis) program being run, which expedites the learning process. This standard display is accomplished because of the existence of the information contained in the templates and data dictionary. When running the REVIEWER, a template is chosen identifying the group of variables to be displayed. The REVIEWER then displays, from the data dictionary, the present value, name, subscript, description, and units for each variable retrieved from the database. Both parametric data and data tables can be contained in the database. The REVIEWER will display data either in list form (Figure 4) or tabular form (Figure 5) as appropriate. The display of variables can be stepped page by page and modified. A subset of a template's variables can be defined to reduce the list of variables displayed. The REVIEWER

is a program separate from the MAKDICT processor; however, the Build REVIEWER Input file (BRV) function of MAKDICT has been designed to expedite the execution of the REVIEWER. The BRV function creates a file containing the information needed to display on a specified template. This avoids the overhead of retrieving the information during the actual REVIEWER process. A detailed discussion of the functions provided by the REVIEWER is presented in the EASIE Volume II [4].

4.5 Executive - Program Execution

The tools described so far, i.e., the ten functions of MAKDICT and the REVIEWER, provide the foundation for efficient integration of analysis programs. This section will describe the execution environment (processor and subprocessors) for conducting multidisciplinary design studies.

The execution environment, as distinguished from the integration environment, is directed primarily to the use of analysis programs and must provide several services. Examples of these services include listing the analysis programs available, reviewing input and output data, selecting a particular version of the database reflecting the model of interest at that time, and designing sequences of program runs that accomplish a specific task. Whereas the integration environment (Figure 6) provides tools to integrate programs [4], the execution environment (Figure 7) provides tools to define and execute design studies [5]. The engineer using the executive need not know anything about the initial integration process.

The execution processor is both menu and command driven. Menus assist the beginner while commands are more efficient for the more knowledgeable user. The executive provides functions that track both the past activities of the designer and the state of the current design. For example, the state of the current design activity is summarized by the status area on the executive display (Figure 8) indicating which application program, template, database version (configuration), and design procedure are being used for the current design task. Additionally, these status values serve as the default values for future commands. These values, as well as a log of past commands and other pertinent information, can be retained in what is called a workspace so that a study may be interrupted, the workspace saved, and the study returned to the breakpoint, days or even weeks after the interruption.

In figure 8, the section entitled PERMANENT MENU summarizes the commands that can be executed from any level within the executive. These include a listing of data configurations, templates, workspaces, application programs, or procedures. Functions are provided for executing the following: issuing operating system commands from within EASIE, adding comments to the log file for marking noteworthy events, altering the menu display, and leaving EASIE. The last section, entitled UTILITY SELECTION, lists the major subprocessors (separate menus) provided by the executive.

A typical design activity, whether computer-aided or not, involves several identifiable steps. The first is definition of engineering data descriptive of an initial model of the system

under study, whether it be a structural analysis model (e.g., geometry or material properties) or electrical subsystem design (e.g., power requirements or parts list). The analysis that is applied to the model usually leads in successive steps to an improved design that may be substantially different from the original model. To track these steps, important versions or configurations of the model must be saved. Using EASIE, this is done by making a copy of the database that contains the configuration. Then, future design changes are applied to a copy of the database while leaving the desired benchmark database intact. Thus, numerous versions of a database, descriptive of different design paths, may be developed. Each user is responsible for maintaining his own database. The executive makes available to all users common items such as the application programs, input/output templates, and master databases. A master is a copy-only version of a database used as starting points for design.

Different designers working with the same model may have different interests. For example, an aerodynamicist may pursue wing/body design while a control system designer may be developing control strategies. Each would begin with a copy of an approved master database containing a common configuration, and each would modify his personal database copy as design needs dictated. As the design progresses, some design changes may be sanctioned by the design manager and new master databases established for the next phase in the iterative design process.

Because the sequence of programs used successfully to conduct a study is likely to be applied again to new versions of the database, the EASIE executive processor allows the creation of procedures which contain the commands needed to execute the sequence. These procedures can be built directly by entering a line for each command, or they can be derived from the activity log (Figure 9) which is kept during operation of the executive. Thus, a designer, who has by trial and error arrived at an effective design sequence, can review the log and select the steps to repeat the desired sequence of commands and then produce a procedure directly from the log. The log also serves as a means to track the design as it develops and thereby can help resolve questions that might arise at a later time.

The procedure-building capability provided by the executive can be used to create very sophisticated sequences which can issue prompts or menus and do conditional branching based on the responses. These automatic procedures will often be created by experienced engineers who wish to establish specific design steps to be followed by a less-experienced person. Thus, certain question and answer scenarios can be built into a procedure to guide a novice.

5.0 THE EASIE PROCESS

A typical scenario can now be described which summarizes the application of the EASIE functions to create and use a new integrated or interfaced system of programs. The function codes in tables I and II will be used to represent the functions performed in the integration and executive processors.

The first step in the EASIE process is for the program experts to define the data dictionary information (all inputs and outputs needed by other programs or selected for database storage) by completing the EASIE coding form. The second step is to enter this information into the database using the integration functions BR, AR, LR, and PR to define the data dictionary including the appropriate grouping or relations. Next, the actual database schema will be produced using the function code BS. Each program will be related to the database information by creating a template for its input list and its output list using functions BT, LT, and PT. The actual connection between the programs and the database can now be made using the function BF to generate subroutines to read from and write to the database. The input subroutines must be added to the application programs replacing any previous input method such as input files or namelist I/O. This requires a manual operation for which guidelines are provided [4]. Output subroutines can co-exist with any previous output methods employed. If no source code is available for the program being added to the system, then interfacing rather than integrating will be required. The subroutines generated by EASIE can be used in the pre- and post-

processing programs. To enable the use of the REVIEWER, the BRV function must be executed to generate the file of database addresses for each variable.

Throughout this process the EASIE interactive processors prompt for descriptions on each element (template, program) created, and this information provides help to the eventual users. The steps just described may take a few hours for a simple system or months for a complex system which requires a great deal of preparation by the program experts and project manager to define the data dictionary and agree on the appropriate way to group variables. In either case, the EASIE tools provide an efficient and systematic means of producing the needed software elements and tracking the buildup of the application system with the variables, programs, and directories.

Once these steps have been completed, the system will be exercised using the executive functions, and a master database will be created which is representative of some baseline configuration with all default values defined. This read-only database will be the basis for standard test cases whenever computer operating system or EASIE software is updated and will be the point of departure for individual analysts to begin their optimization or trade studies. The experienced analyst may choose the functions PBLD and PREX to develop his own procedures or to create for junior engineers some predetermined program execution procedure which has a very narrow study focus. Typically, the APEX, DATA, TBLD, and WSC functions will be used to select programs, edit data, review output, and maintain databases and work progress.

6.0 CURRENT APPLICATIONS

The EASIE software serves, to various degrees, as the framework for several developing design systems at LaRC. The degree to which EASIE has been used depends primarily on the maturity of these tools at the time the design system was being constructed.

The Aerospace Vehicle Interactive Design (AVID) System [1], for conceptual design of launch vehicles, is depicted in Figure 10. This figure represents more of a goal than the current status of the system. Several pieces have been interfaced, but in fact, the difficulties originally encountered in this task formed the basis for the EASIE specifications. Now that EASIE is operational, activities are underway to complete the AVID system.

The LaRC Interactive Design and Evaluation of Advanced Spacecraft (IDEAS) System (Figure 11) was developed initially using early vintage integration tools, but it has now been implemented in EASIE using the integration processor. That is, the database has been constructed and programs integrated for data exchange via EASIE, but the executive has not been applied. This is because the original IDEAS programs were all highly interactive, and the benefits derived by conversion to the REVIEWER could not outweigh the labor required. However, new programs implemented into the system will anticipate using the full capabilities of EASIE. As a side note, it should be stated that within most integrated systems and with EASIE in particular, batch programs are more desirable and efficient than interactive programs.

The previous two examples, AVID and IDEAS, represent a substantial investment in analysis codes and a significant requirement for maintenance and improvement. This latter fact has led to the acquisition of commercial software wherever the capabilities and reliability of such software will support the LaRC design needs. This presents a new problem in that locally developed software, which allows a fast response to specific design issues, must be somehow coordinated with a vendor's product. This problem was addressed when early space station study requirements at LaRC led to the acquisition of the Structural Dynamics Research Corporation (SDRC) Integrated Design and Engineering Analysis Software (I-DEAS®) [6]. It was desirable to coordinate several programs from the NASA IDEAS system with the solid modeling and finite element modeling tools within SDRC's I-DEAS. The product that resulted, known as IDEAS**2 (IDEAS Squared), has been implemented by using the EASIE utilities to attach the NASA programs to the commercial I-DEAS package. The advantage to such an approach is that LaRC can exercise some latitude to expand the number of locally developed programs integrated into the system and thereby achieve study goals in a short time without waiting for the marketplace to warrant the commercial development of each analysis code required. The IDEAS**2 software is being used by the Space Station Program at Reston, Virginia, and versions of it exist also at LaRC and the Johnson Space Center.

® I-DEAS is a trademark of Structural Dynamics Research Corp.

7.0 COMPUTERS AND SOFTWARE

EASIE, Version 1.0, was developed on a VAX 11/785 under the VMS operating system and uses the RIM software as its DBMS. All of the EASIE software is written in FORTRAN. The data dictionary assumes variables to be FORTRAN and will ask for certain FORTRAN characteristics such as variable type (real, integer, character) and array dimensions. Also, the FORMATTER produces FORTRAN subroutine code for accessing the database variables. There is no reason inherent in the EASIE design that requires FORTRAN; however, this version's implementation reflects the fact that the analysis programs and DBMS of interest were all FORTRAN.

Likewise, the architecture of the EASIE user interfaces has been designed, as much as possible, to be independent of the DBMS software. The database management system used initially was A Relational Information System (ARIS) developed at LaRC. However, the need for more widely available DBMS software led to the selection of RIM [3]. Currently, many of the EASIE tools have been used to add capabilites to the NASA IDEAS**2 system. Thus, capabilities exist to support the Project Relational Language (PRL) of the SDRC I-DEAS.

The unavoidable need to organize the programs, templates, and data areas that accompany an architecture such as EASIE's has led to the definition of a particular hierarchical file structure on the VAX/VMS system. This structure [7], although not required, has provided a very efficient and readable framework for locating software components as they are integrated. In addition, these structures also assist the installation of the EASIE system onto a new host computer.

As a final note, the existence of RIM on the CDC Cyber series of computers has opened the possibility of placing EASIE on the Cybers. An option, not fully tested, has been provided to handle the word length differences between the Cyber architecture and that of PRIME or VAX. Greater details on the consequences of changing either the host computer or the embedded DBMS are provided in EASIE Version 1.0, Volume IV, System Installation and Maintenance Guide [8].

8.0 SUMMARY

The EASIE utilities provide a systematic method of developing coordinated systems of application programs, via interfacing or integration, that exchange data through a central database. Utilities exist for conducting the necessary function of the integration tasks and the execution tasks. The software has been documented and is operational in several LaRC-developed design systems for launch vehicles and spacecraft and has been used to extend the analysis capabilities of one commercial product. The recent applications of the EASIE utilities has revealed areas where improvements can be made in user-friendliness and functionality, and a version 2.0 has been defined and is currently under development.

Function Code	Function Performed
BR	Build relations
AR	Add variables to a relation
LR	List relations
PR	Print a relation
BS	Build a database schema dump
BT	Build a template
LT	List a template
PT	Print a template
BF	Build FORMATTER subroutines for reading and writing the database
BRV	Build REVIEWER input file

Table I - EASIE Integration Processor Functions.

Function Code	Function Performed
APEX	Application Program Execution
DATA	Data Modification
PBLD	Procedure Building
PREX	Procedure Execution
TBLD	Template Building
WSC	Workspace Control

Table II - EASIE Execution Processor Functions.

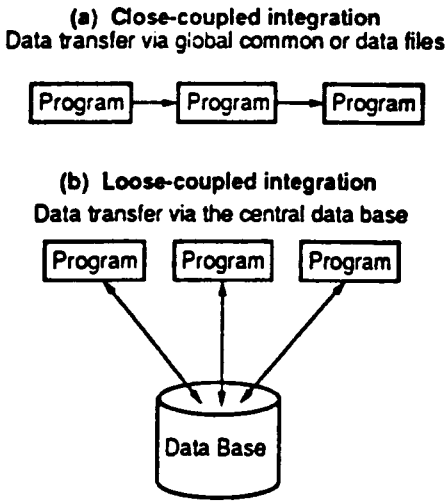

Figure 1. - Program Integration Techniques

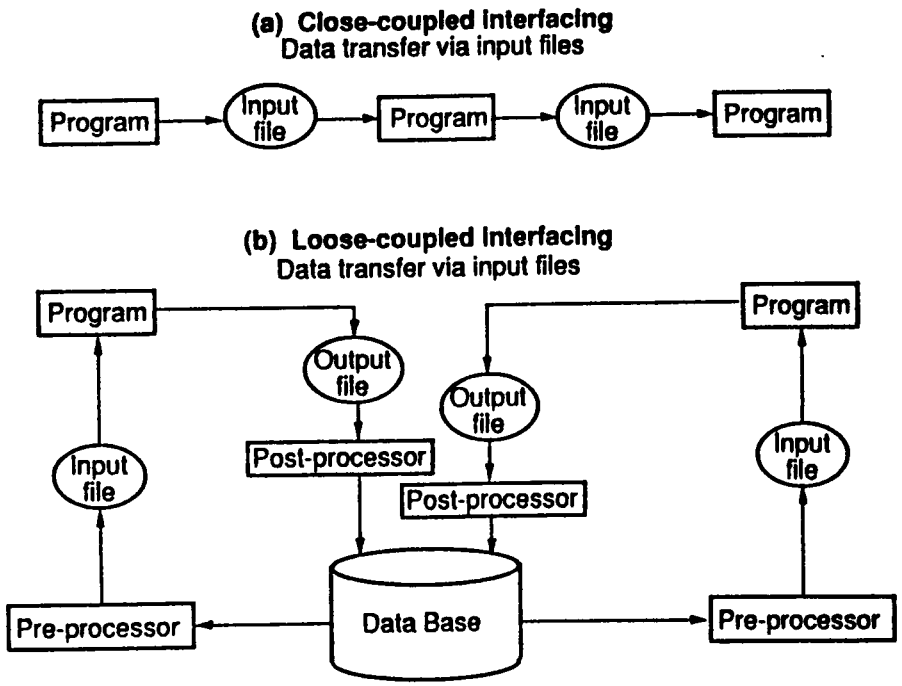

Figure 2. - Program Interfacing Techniques

EASIE CODING FORM	VARIABLE DEFINITION AND USE	PAGE OF PAGES	DATE
DESIGN SYSTEM	APPLICATION PROGRAM	CONTACT PERSON PHONE	

1	2	3	4	5	6	7	8	9	10	11	12	13
REFERENCE DOCUMENT SYMBOL	VARIABLE NAME (8 CHAR.)	TYPE*	ARRAY DIMENSION (X,X,X)	VARIABLE DESCRIPTION (80 CHARACTERS)* (First 16 characters should be most descriptive)	UNITS* (16 CHAR.)	DISPLAY FORMAT (16 CHAR.)	DEFAULT VALUE	USE*	COMMON BLOCK NAME	ATTRIBUTE/ PARAMETER NAME	RELATION NAME (8 CHAR.)	TYPE

NASA EASIE FORM 1
JULY 1987

* See Instructions for restrictions

SSD - SAB E - 1

Figure 3. - EASIE Coding Form

ORIGINAL PAGE IS
OF POOR QUALITY

```
                                CATEGORY      1:    MODEL
MODEL INFORMATION

-------------------------------------------------------------------------------
 L  !   PRESENT VALUE  !   NAME    ! SUBSCRIPT  ! DESCRIPTION      ! UNITS
-------------------------------------------------------------------------------
  1 ! TEST OF MAKGEO F ! NAME      !            ! MODEL NAME       !
  2 !       20.0000000 ! ROTATION  !         1  ! MODEL X,Y,Z ROTA ! DEGREES
  3 !       30.0000000 !           !         2  !                  !
  4 !       40.0000000 !           !         3  !                  !
-------------------------------------------------------------------------------

M n : modify value (n = line#,name(subscript),or line range)
C n : change category (n = id or name)
N n : next page (n = + or - pages)
R : reprint page, L n : n line#'s per page, X n : expand line# n
E : end and save mods, Q : quit without saving mods, H : help
CAT : list categories, SUB : define review subset, T : toggle menu
EDIT:
>
```

Figure 4. - REVIEWER Display of Parameter Data

```
                                CATEGORY      3:    FACES
CONNECTIVITY OF NODES TO FORM FACES

-----------------------------------------------------
NAME   !  FACE     !  FACE     !  FACE     !  FACE

INDEX  !        1  !        2  !        3  !        4

-----------------------------------------------------
COL    !       1   !       2   !       3   !       4

-----------------------------------------------------
ROW :
      1!         1 !         2 !         3 !         4
      2!         2 !         6 !         7 !         3
      3!         6 !         5 !         8 !         7
      4!         5 !         1 !         4 !         8
      5!         4 !         3 !         7 !         8
      6!         5 !         6 !         2 !         1
-----------------------------------------------------

M r c : modify value (r = row# or range; c = column#,name(subscript), or range)
C n : change category (n = id or name)
N n : next page (n = + or - pages)
R : reprint page, L n : n rows per page, X r c : expand row# n, column# c
S : set columns to be displayed
E : end and save mods, Q : quit without saving mods, H : help
CAT : list categories, SUB : define review subset, T : toggle menu
EDIT:
>
```

Figure 5. - REVIEWER Display of Attribute Data

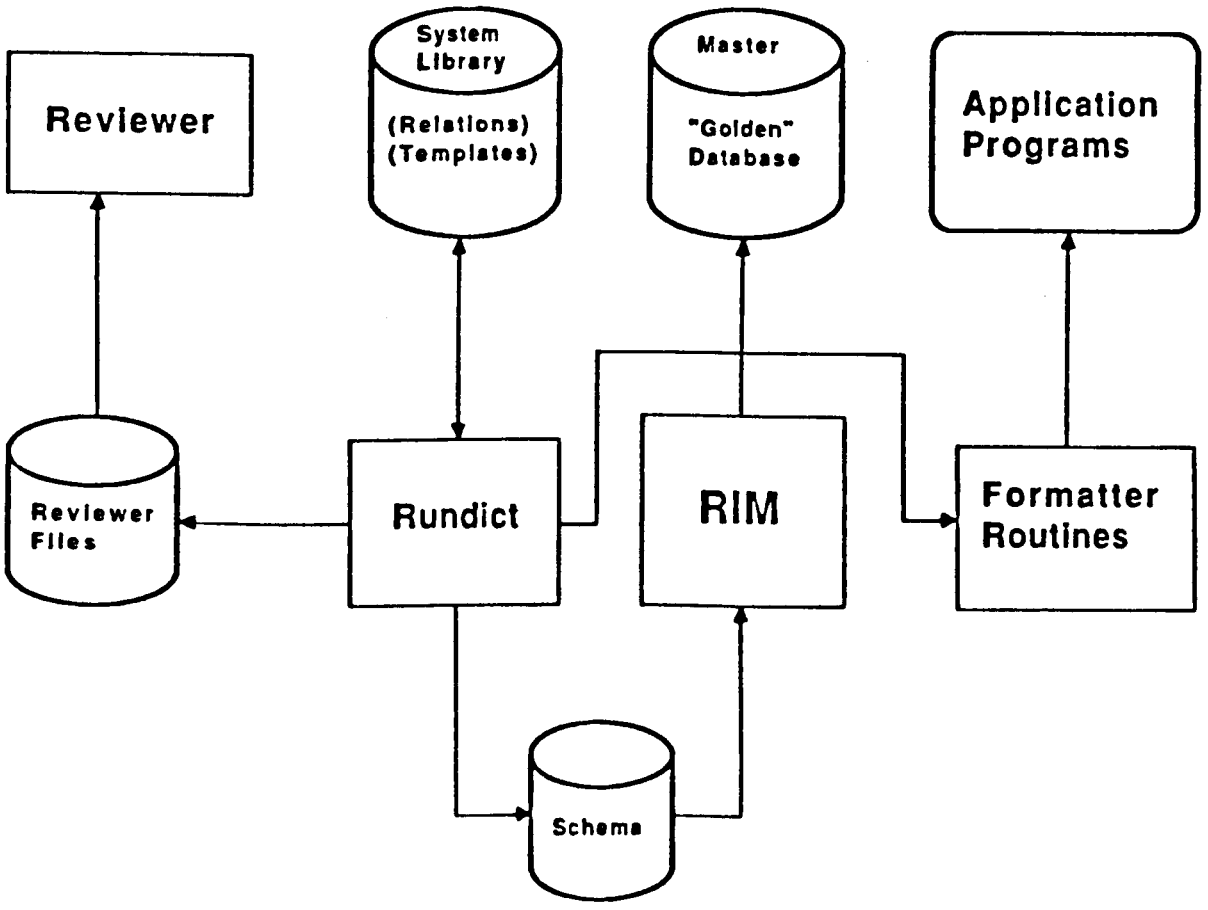

Figure 6. - The Integration Environment

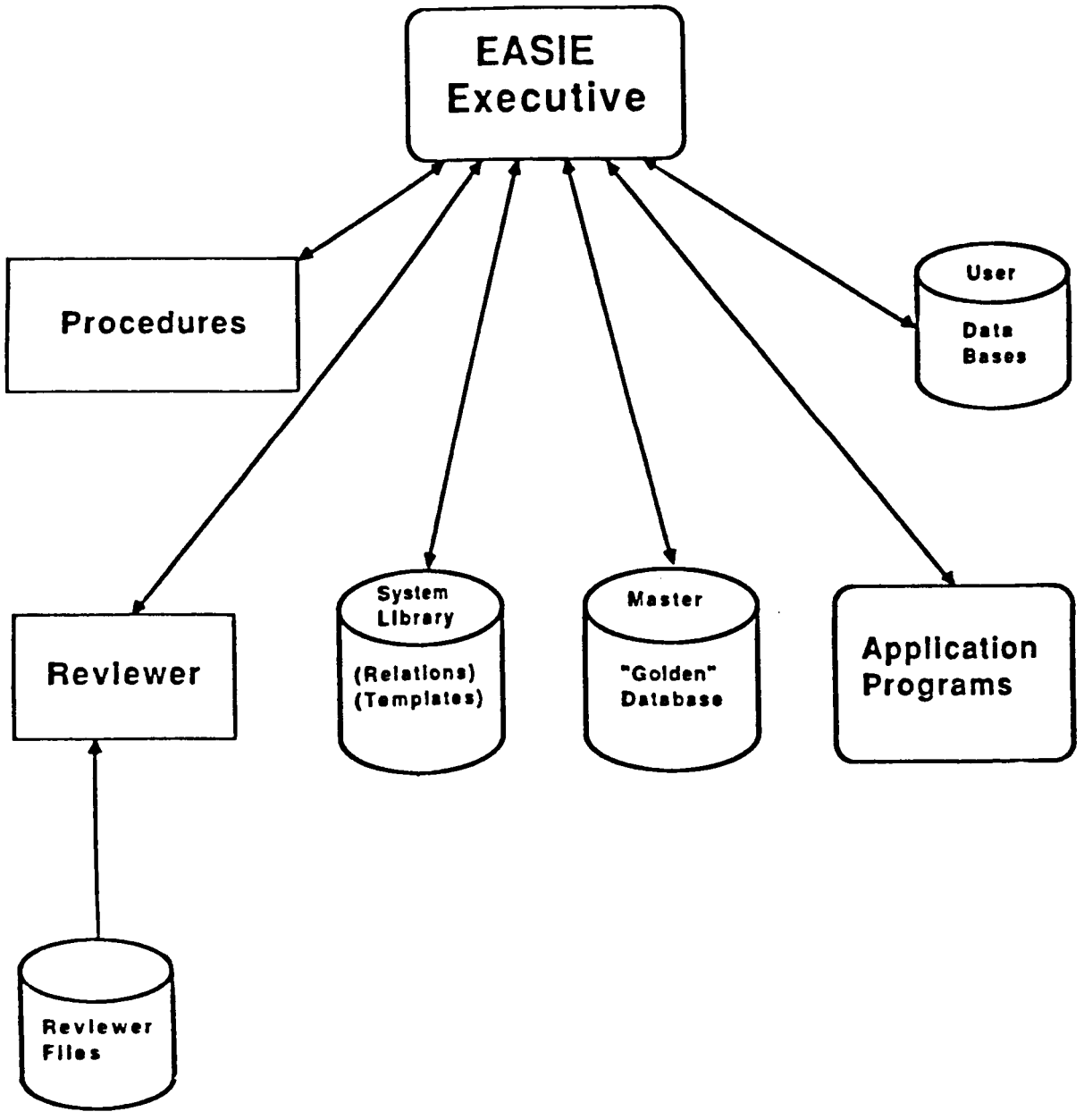

Figure 7. - The Execution Environment

"MAIN MENU"

```
          STATUS:
                                    WORKSPACE:     T$            REF_CONFIG:
UTIL_IN_USE:   UTILITY SELECTI
                                    APPLIC_PROG:                 REF_TEMPLATE:
REF_PROCFILE:

          PERMANENT MENU:

          H - HELP
          D - DIRECTORY of CFG's, TPL's and WS's
          I - INVENTORY of APPL's and PROC's
          S - SYSTEM COMMANDS for PRIMOS
          C - Add a COMMENT to the command log
          T - TOGGLE the MENU PRINT setting
          R - RETURN to the PREVIOUS MENU
          Q - QUIT this sequence of menus and
              RETURN to the MAIN MENU
          L - LOGOUT

          <CR> - Clear the screen and relist the menu

          UTILITY SELECTION (MAIN)

                                          COMMAND FORMAT
          1 - WORKSPACE CONTROL UTILITY          ACT UTL  <WSC >
          2 - DATA MODIFICATION UTILITY          ACT UTL  <DATA>
          3 - APPLICATION EXECUTION UTILITY      ACT UTL  <APEX>
          4 - PROCEDURE EXECUTION UTILITY        ACT UTL  <PREX>
          5 - PROCEDURE BUILDING UTILITY         ACT UTL  <PRLD>
          6 - TEMPLATE BUILDING UTILITY          ACT UTL  <TBLD>

        ENTER COMMAND:
```

Figure 8. - Main Executive Menu

```
 WS ACTIVATED ON :  4-DEC-87
>ACT UTL  WSC
>T
* TOGGLE TTY PRINT MODE
>D
* DIRECTORY
>I
* INVENTORY
>CP CFG DEFAULT NUDATA
* COPY FROM:TOAIDE:[EXAMPLE.CFG.DEFAULT]          TO:NUDATA
• CARRIED OUT COMMAND: N CFG
>ACT APPL BOX
>RVU IDB
>Q
• QUIT THIS MENU, RETURN TO MAIN
>ACT UTL  APEX
>EX APPL BOX
*    EXECUTING BOX
>RVU ODB
>TY LOG   T$RTA4
```

Figure 9. - Example Log File

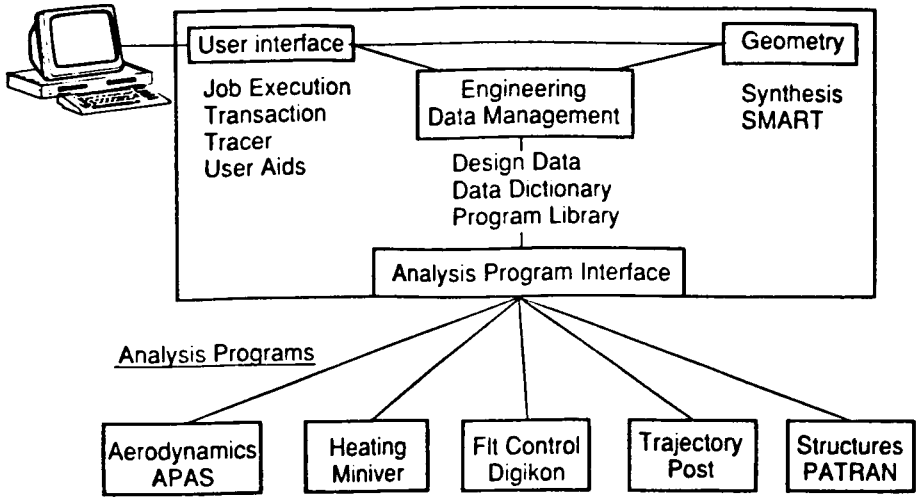

Figure 10. - Concept for the Aerospace Vehicle Interactive Design (AVID) System

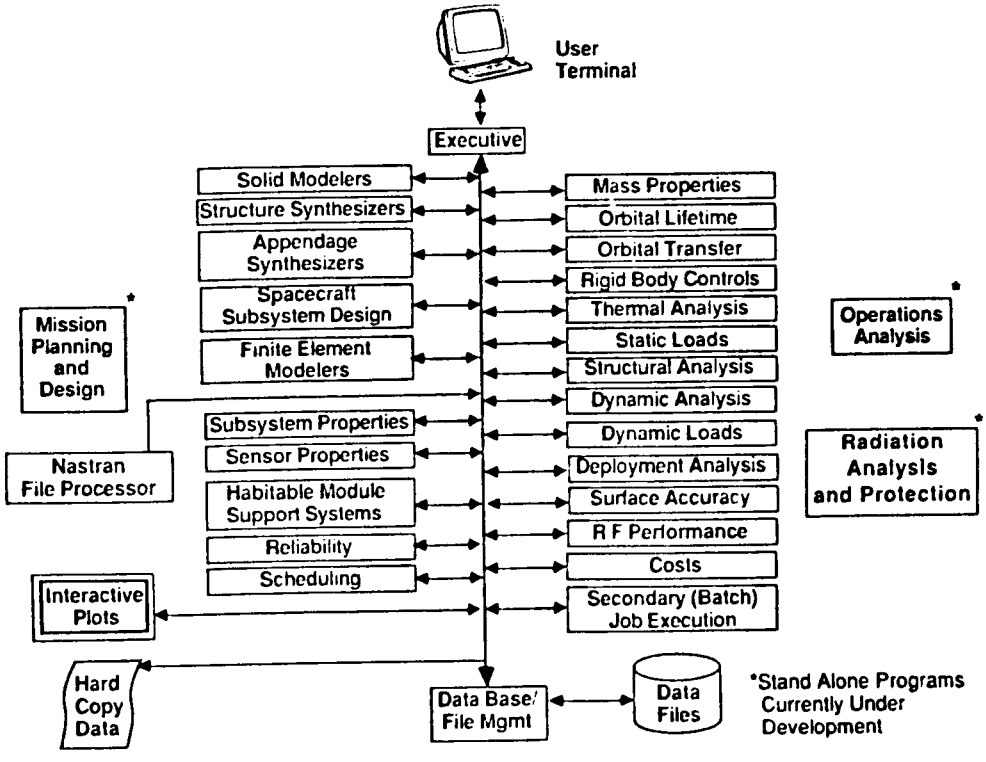

Figure 11. - The Interactive Design and Evaluation of Advanced Spacecraft (IDEAS) System

ORIGINAL PAGE IS
OF POOR QUALITY

REFERENCES

1. Wilhite, A,; Johnson, S. C.; and Crisp, V.: "Integrating Computer Programs For Engineering Analysis and Design," AIAA Paper No. 83-0597, 21st Aerospace Sciences Meeting, January 1983.

2. Date, C. J.: The Systems Programming Series Volume I - AN INTRODUCTION TO DATABASE SYSTEMS, Third Edition, Addison-Wesley Publishing Company, February 1982.

3. Boeing Computer Services Company, "BCS RIM - Relational Information Management System Version 6.0 User Guide, May 1983.

4. Jones, Kennie H.; Randall, Donald P.; Stallcup, Scott S. and Rowell, Lawrence F.: The Environment For Application Software Integration and Execution (EASIE) Version 1.0. VOLUME II - PROGRAM INTEGRATION GUIDE. NASA TM-100574, April 1988.

5. Schwing, J. L; Rowell, L. F.; and Criste, R. E.: The Environment For Application Software Integration and Execution (EASIE) Version 1.0 - Volume III - PROGRAM EXECUTION GUIDE," NASA TM-100575, April 1988.

6. Structural Dynamics Research Corporation, I-DEAS™ USER'S GUIDE, 1986.

7. Dube, R. P.; and Smith, M. R.: "Managing Geometric Information With A Database Management System", IEEE Computer Graphics and Applications, V. 3, No. 7, pp. 57-62, October 1983.

8. Randall, Donald P.; Jones, Kennie H.; and Rowell, Lawrence F.: The Environment For Application Software Integration and Execution (EASIE) Version 1.0. VOLUME IV - SYSTEM INSTALLATION AND MAINTENANCE GUIDE. NASA TM-100576, April 1988.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-100573		

4. Title and Subtitle
The Environment For Application Software Integration and Execution (EASIE) Version 1.0, Volume I - Executive Overview

5. Report Date
March 1989

6. Performing Organization Code

7. Author(s)
Lawrence F. Rowell and John S. Davis

8. Performing Organization Report No.

10. Work Unit No.
506-49-31-01

9. Performing Organization Name and Address
NASA Langley Research Center
Hampton, VA 23665-5225

11. Contract or Grant No.

13. Type of Report and Period Covered
Technical Memorandum

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, DC 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes
Lawrence F. Rowell, Langley Research Center, Hampton, VA
John S. Davis, Computer Sciences Corporation, Hampton, VA

16. Abstract

The Environment For Application Software Integration and Execution, EASIE, provides a methodology and a set of software utility programs to ease the task of coordinating engineering design and analysis codes. EASIE was designed to meet the needs of conceptual design engineers that face the task of integrating many stand-alone engineering analysis programs. Using EASIE, programs are integrated through a relational database management system.

Volume I, Executive Overview, gives an overview of the functions provided by EASIE and describes their use. Three operational design systems based upon the EASIE software are briefly described.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
EASIE Program Interfacing Program Integration Database Management Data Dictionary	Unclassified - Unlimited Subject Category - 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	40	A03